Redis

Redis简介

Redis是一个基于内存的key–value（键值对）结构数据库

- 基于内存存储，读写性能高
- 访问量大的适合存储热点数据（热点商品、咨询、新闻）
- 企业应用广泛

官网：**http://redis.io**

中文网站：http://www.redis.net.cn/

Redis下载与安装

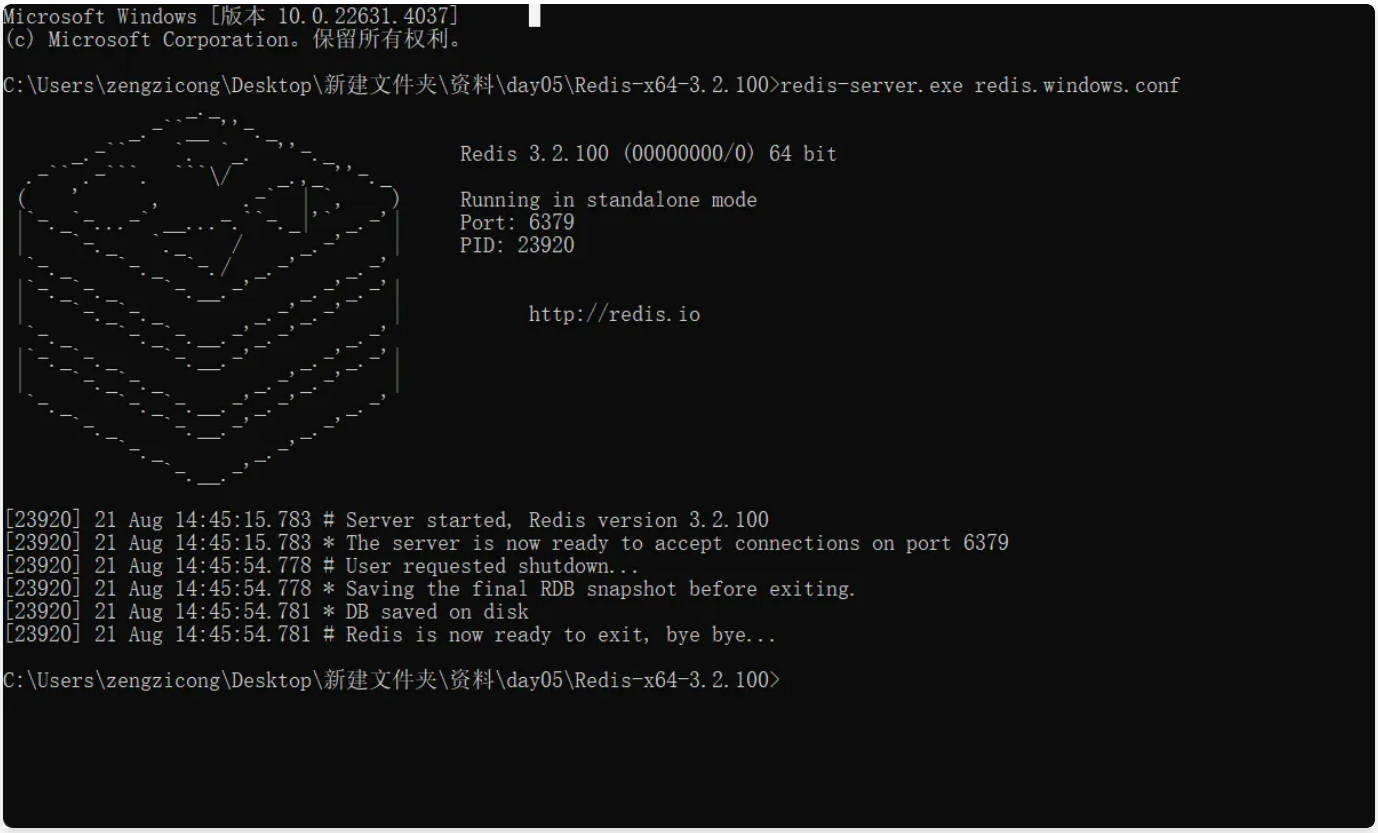
window下载地址：https://github.com/microsoftarchive/redis/releases

EventLog.dll	2016/7/1 16:27	应用程序扩展	1 KB
Redis on Windows Release Notes.docx	2016/7/1 16:07	DOCX 文档	13 KB
Redis on Windows.docx	2016/7/1 16:07	DOCX 文档	17 KB
redis.windows.conf redis配置文件	2016/7/1 16:07	CONF 文件	48 KB
redis.windows-service.conf	2016/7/1 16:07	CONF 文件	48 KB
redis-benchmark.exe	2016/7/1 16:28	应用程序	400 KB
redis-benchmark.pdb	2016/7/1 16:28	PDB 文件	4,268 KB
redis-check-aof.exe	2016/7/1 16:28	应用程序	251 KB
redis-check-aof.pdb	2016/7/1 16:28	PDB 文件	3,436 KB
redis-cli.exe redis客户端	2016/7/1 16:28	应用程序	488 KB
redis-cli.pdb	2016/7/1 16:28	PDB 文件	4,420 KB
redis-server.exe redis服务端	2016/7/1 16:28	应用程序	1,628 KB
redis-server.pdb	2016/7/1 16:28	PDB 文件	6,916 KB
Windows Service Documentation.docx	2016/7/1 9:17	DOCX 文档	14 KB

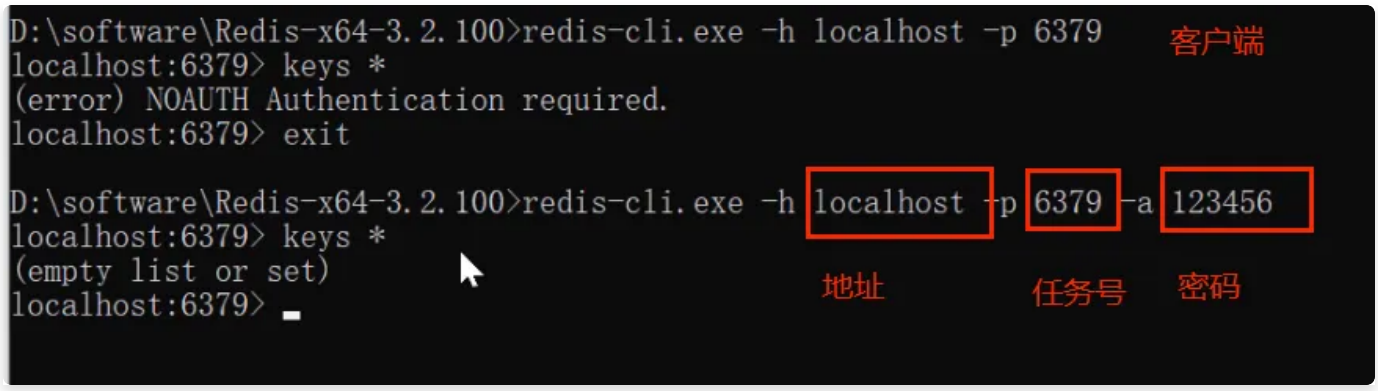
服务端：

```
Microsoft Windows [版本 10.0.22631.4037]
(c) Microsoft Corporation。保留所有权利。

C:\Users\zengzicong\Desktop\新建文件夹\资料\day05\Redis-x64-3.2.100>redis-server.exe redis.windows.conf
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 3.2.100 (00000000/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 23920
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[23920] 21 Aug 14:45:15.783 # Server started, Redis version 3.2.100
[23920] 21 Aug 14:45:15.783 * The server is now ready to accept connections on port 6379
```

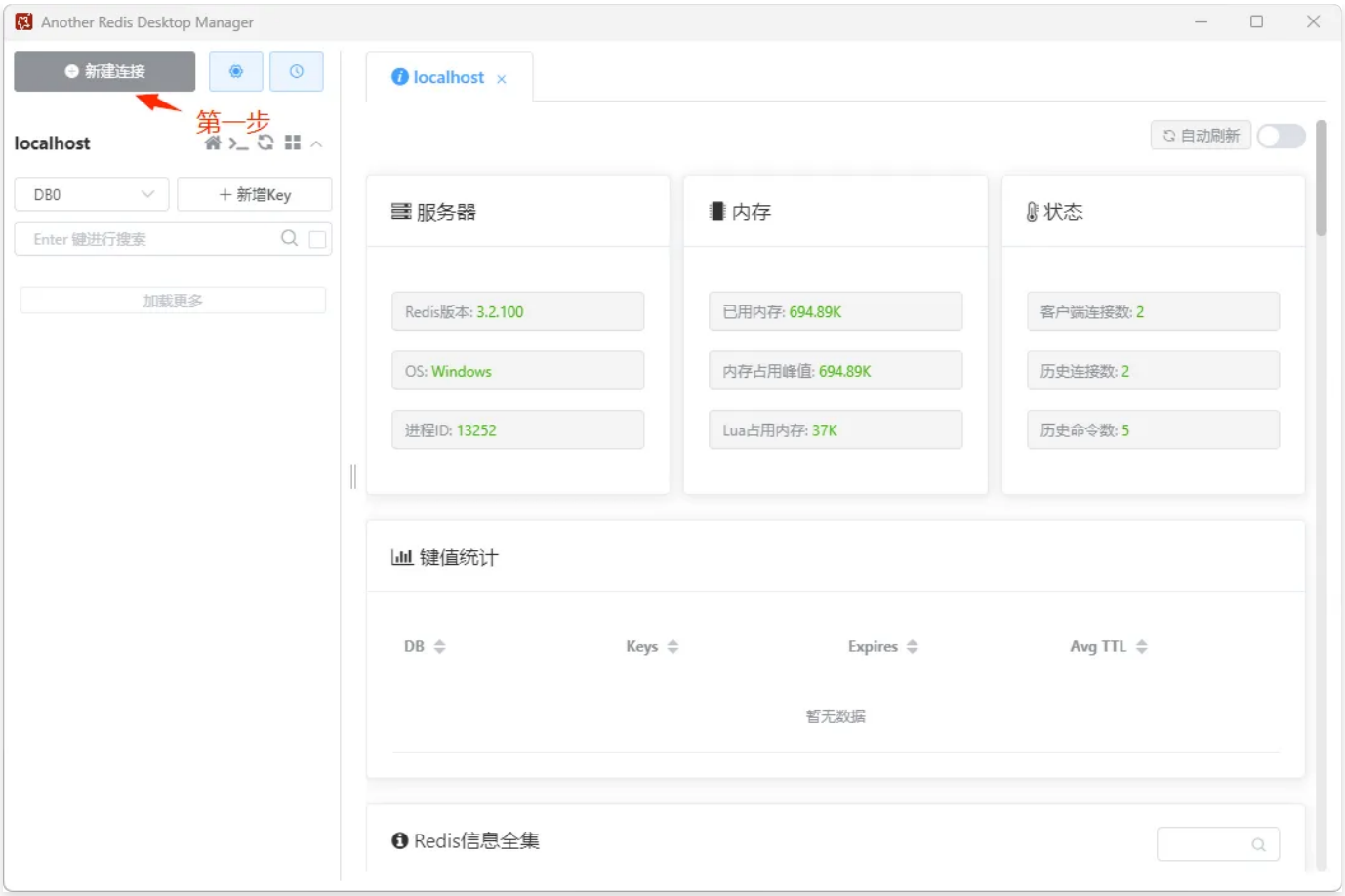
ctrl + c：结束进程



客户端：exit退出



Anthor-Redis-Desktop-Manager安装


Another Redis Desktop Manager
新建连接
第一步
localhost
DB0
+ 新增Key
Enter 键进行搜索
加载更多
localhost
自动刷新
服务器
Redis版本: 3.2.100
OS: Windows
进程ID: 13252
内存
已用内存: 694.89K
内存占用峰值: 694.89K
Lua占用内存: 37K
状态
客户端连接数: 2
历史连接数: 2
历史命令数: 5
键值统计
DB
Keys
Expires
Avg TTL
暂无数据
Redis信息全集


新建连接
localhost
自动刷新
新建连接
* 地址
127.0.0.1
* 端口
6379
密码
用户名
ACL in Redis >= 6.0
连接名称
分隔符
:
SSH
SSL
Sentinel
Cluster
Readonly
只需填这几个就行
取消
确定
Redis信息全集

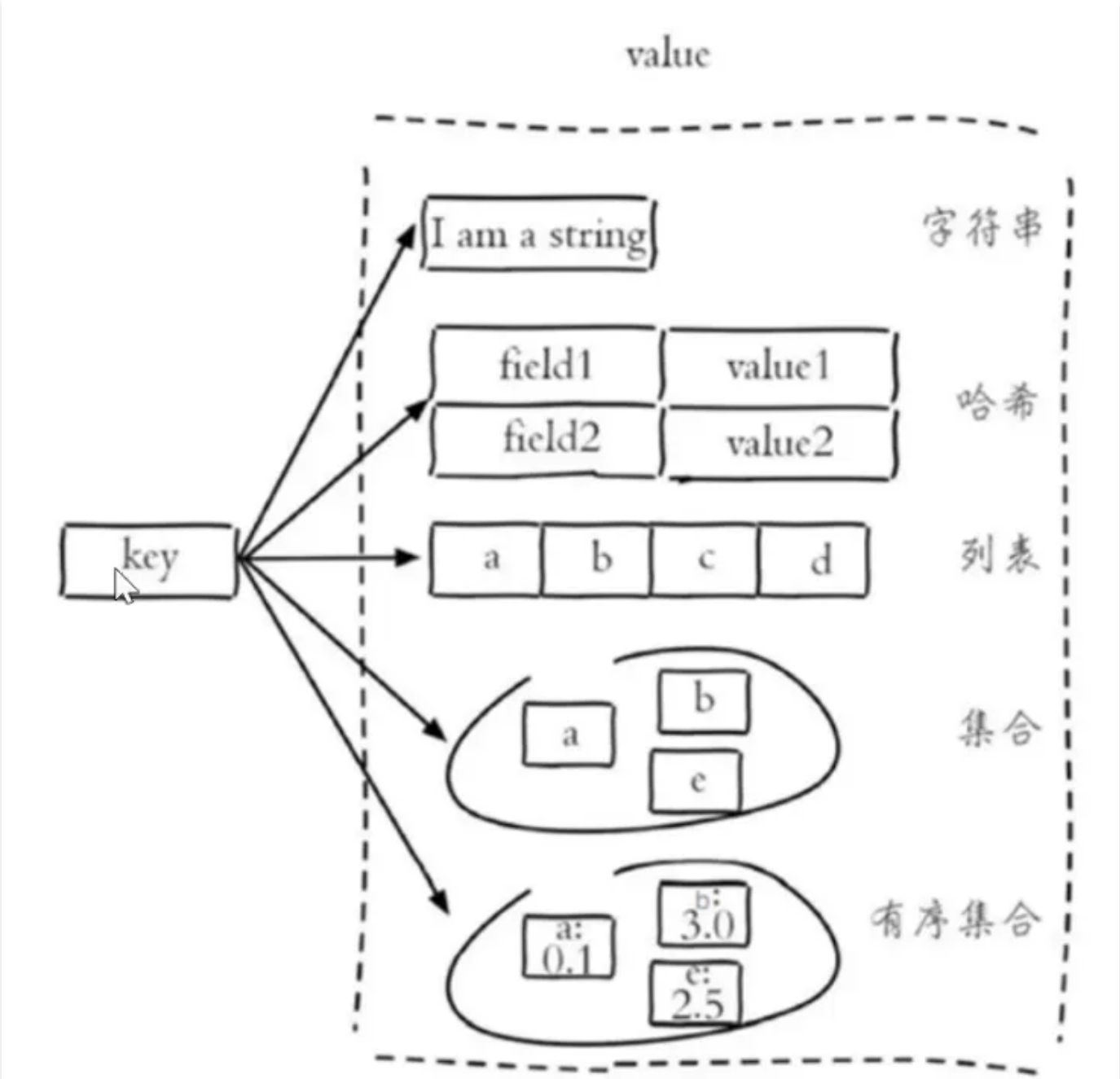
Redis的数据类型

1. 5种常用数据类型

Redis存储的是Key-value结构的数据，其中key是字符串类型，value有5种常用的数据类型：

- 字符串 String
- 哈希 hash
- 列表 list
- 集合 set
- 有序集合 sorted set / zset

2. 各种数据类型的特点



字符串（string）： 普通字符串，redis中最简单的数据类型

哈希（hash）： 也叫散列，类似于java中的HashMap结构

列表（list）： 按照插入顺序排序，可以有重复元素，类似于java中的LinkesList

集合（set）： 无序集合，没有重复元素，类似于java中的Hashset

有序集合（sorted set / zset）： 集合中每个元素关联一个分数（score），根据分数升序排序，没有重复元素

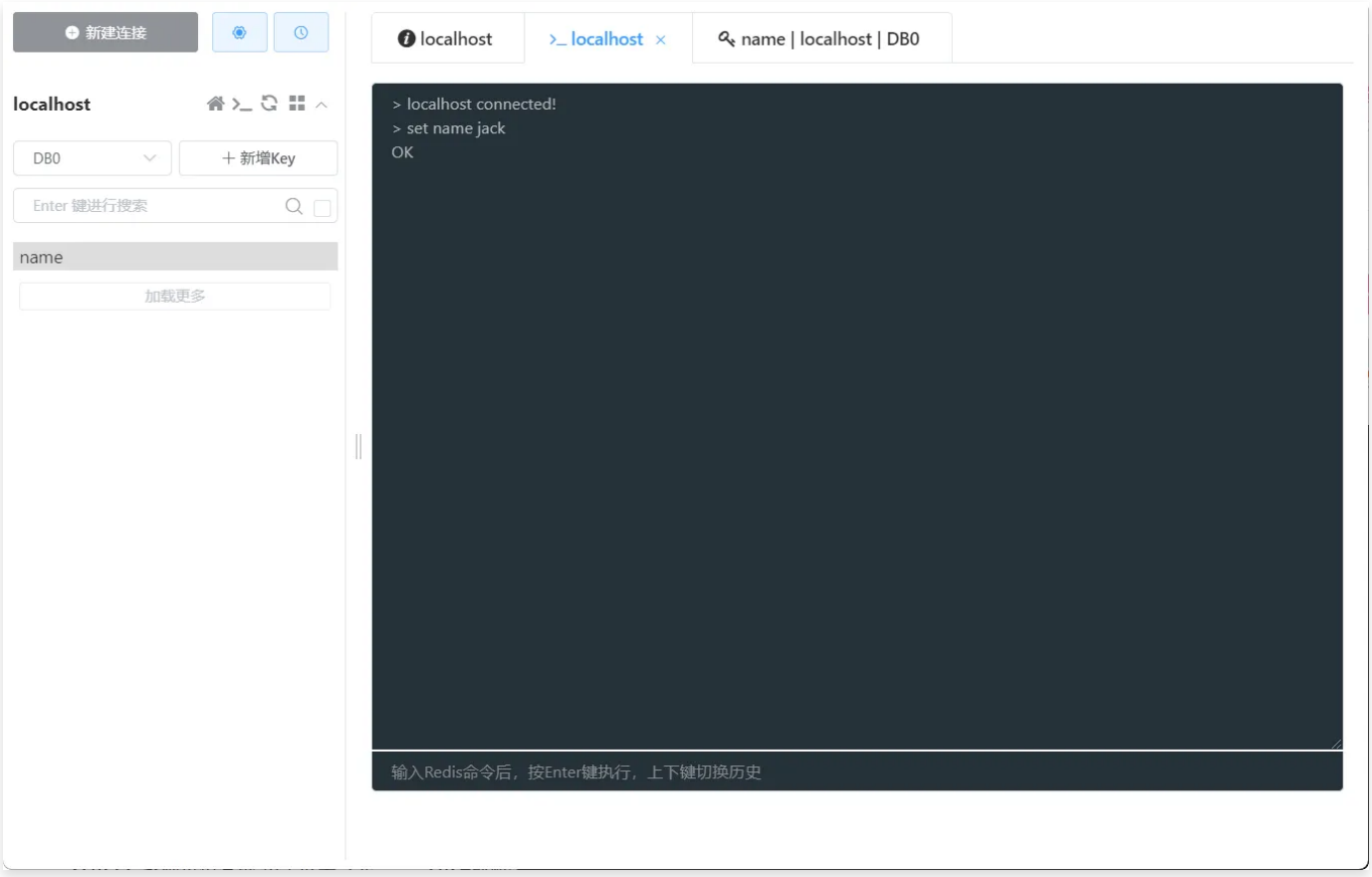
Redis中常用命令

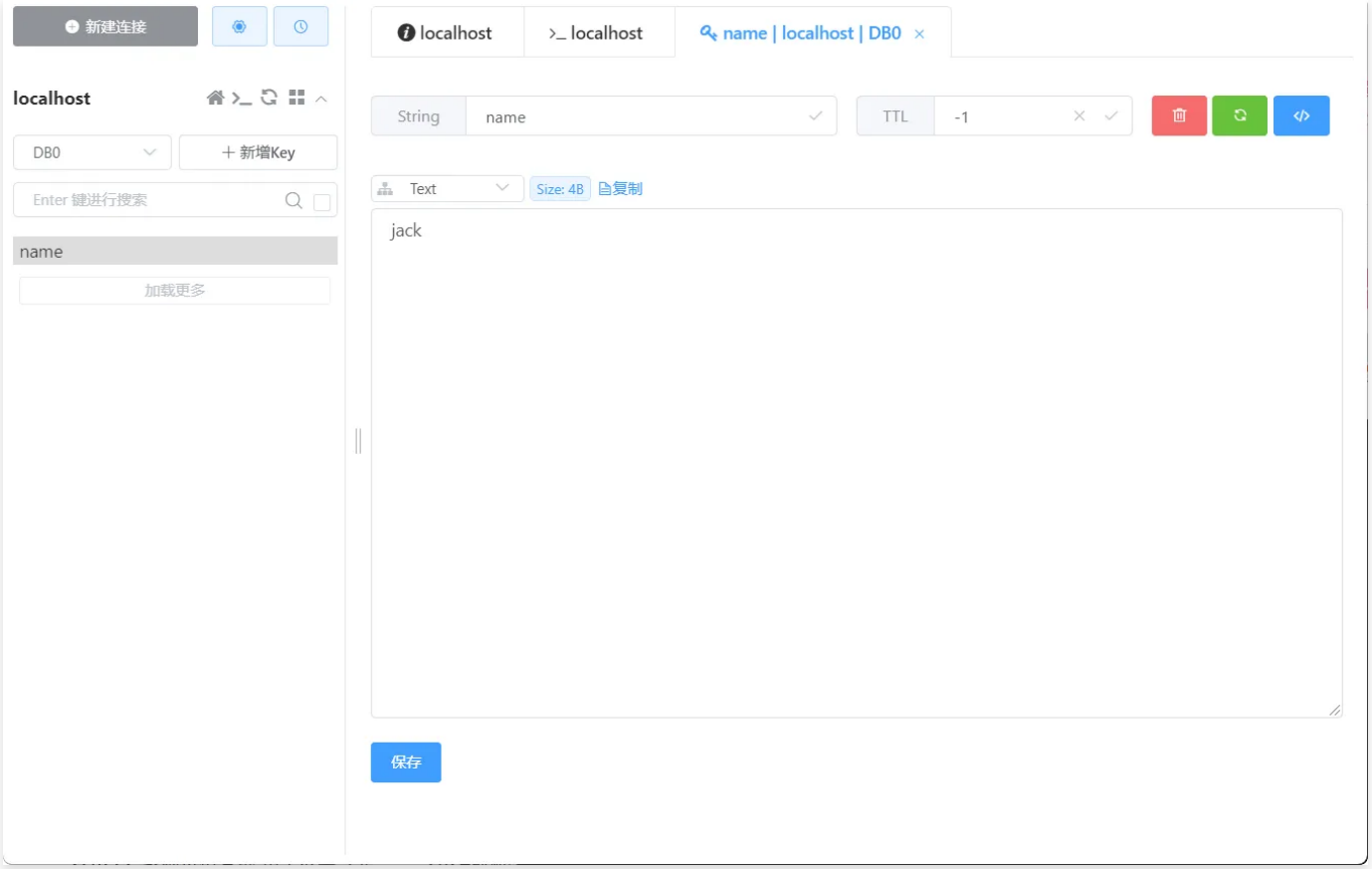
- 字符串操作命令
- 哈希操作命令
- 列表操作命令
- 集合操作命令
- 有序集合操作命令
- 通用操作命令

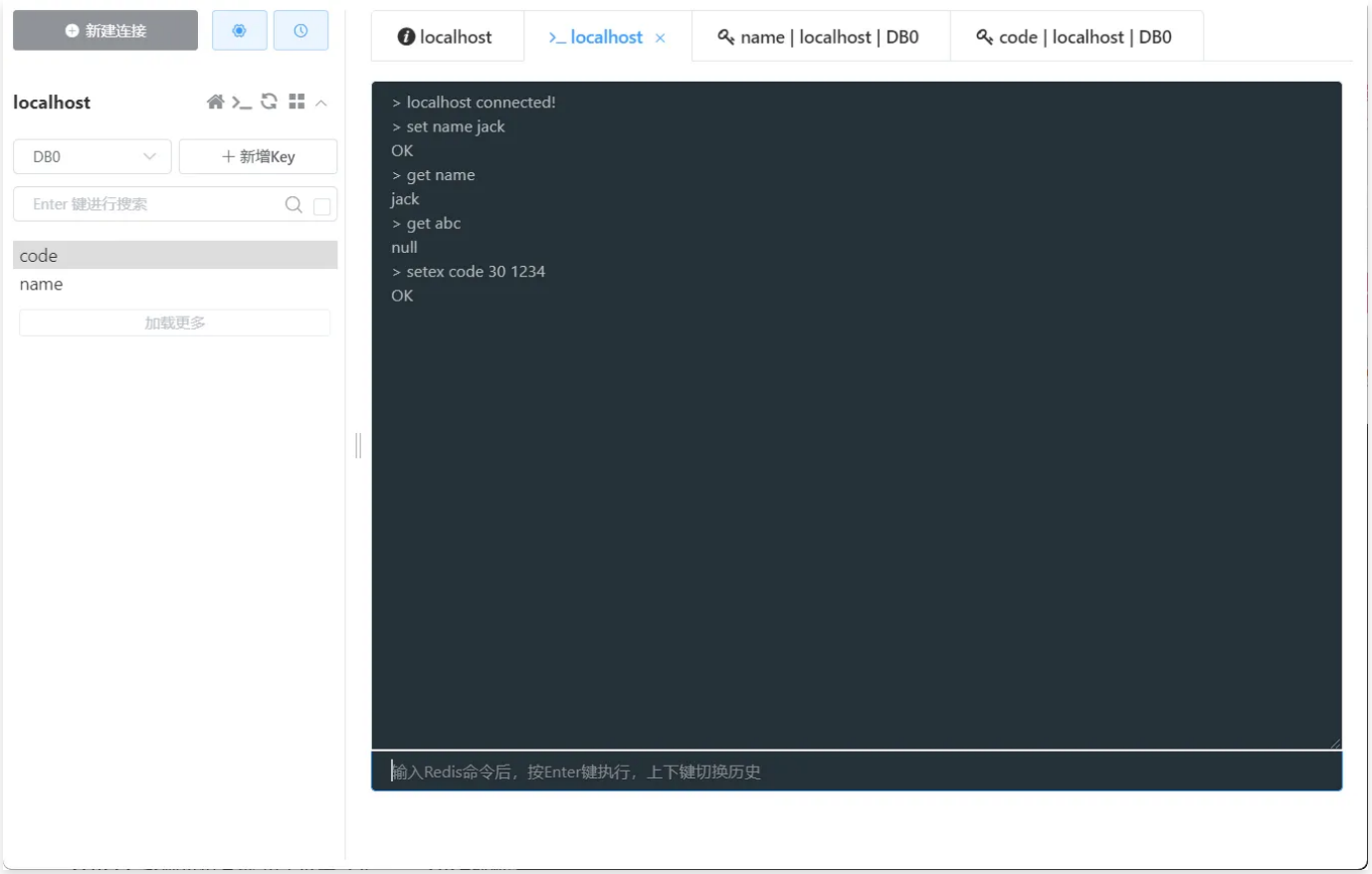
字符串操作命令

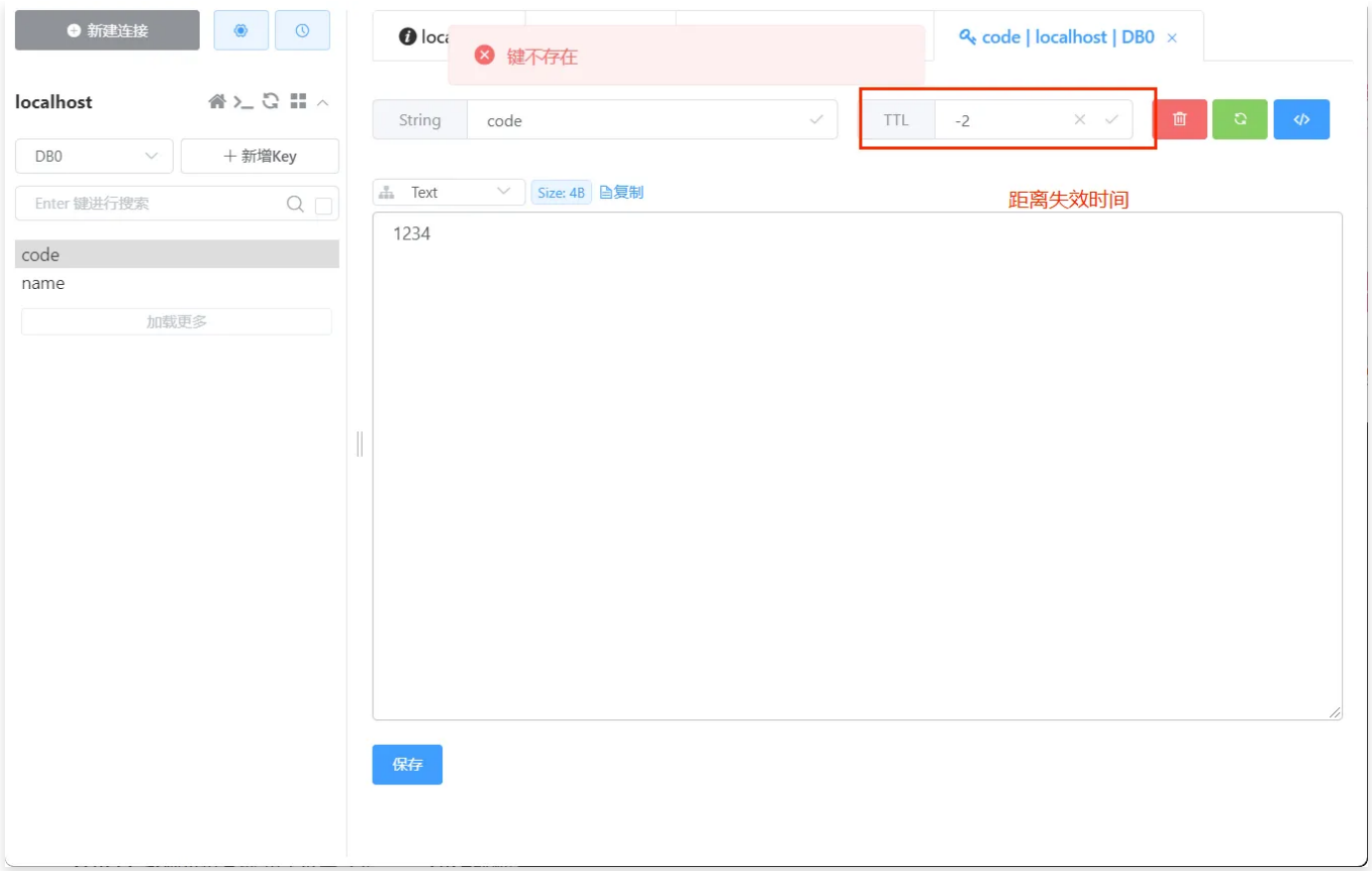
Redis字符串类型常用命令：

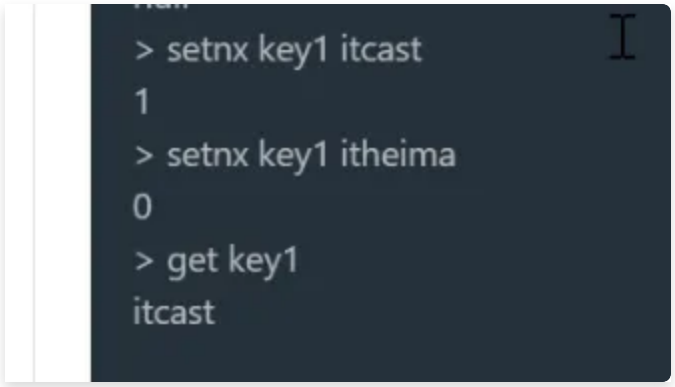
- SET key value　　设置指定key的值
- GET key　　获取指定key的值
- SETEX key seconds value　　设置指定key的值，并将key的过期时间设置为seconds秒，会自动删除
- SETNX key value　　只有在key不存在时设置key的值




新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
name
加载更多
localhost
>_ localhost
name | localhost | DB0 ×
String
name
TTL
-1
Text
Size: 4B
复制
jack
保存


新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
code
name
加载更多
localhost
>_ localhost ×
name | localhost | DB0
code | localhost | DB0
> localhost connected!
> set name jack
OK
> get name
jack
> get abc
null
> setex code 30 1234
OK
输入Redis命令后，按Enter键执行，上下键切换历史

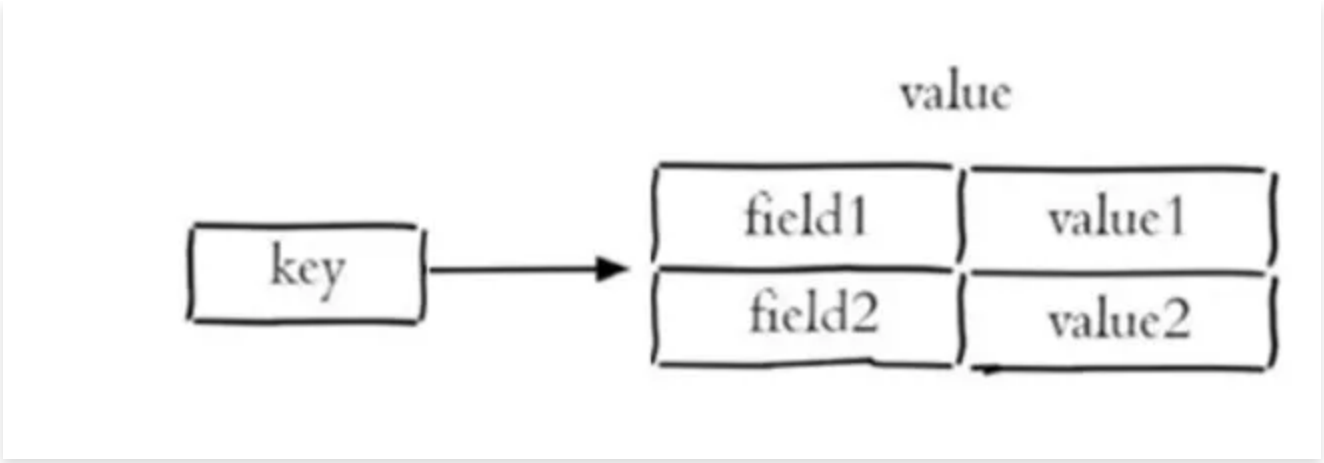


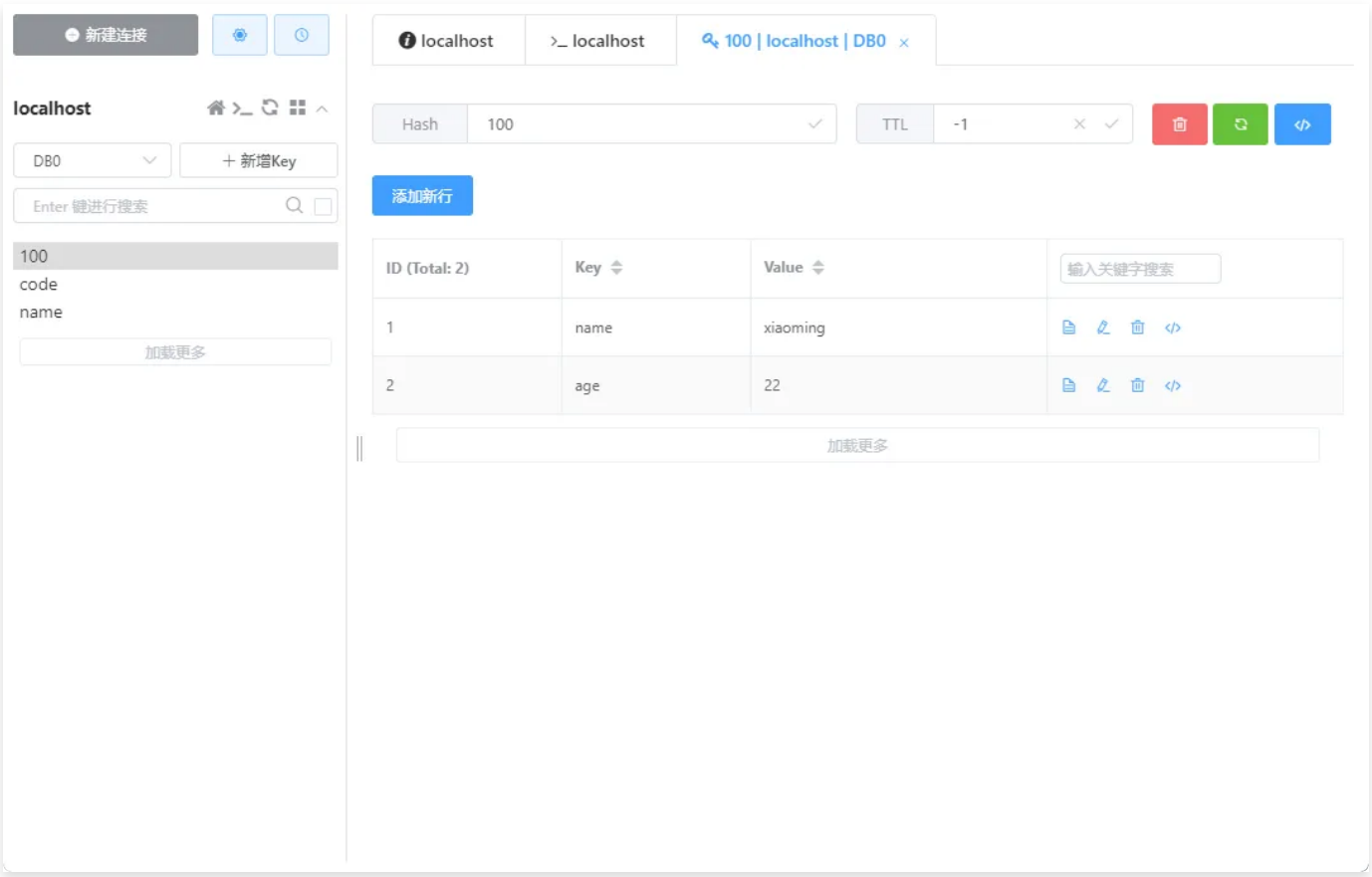


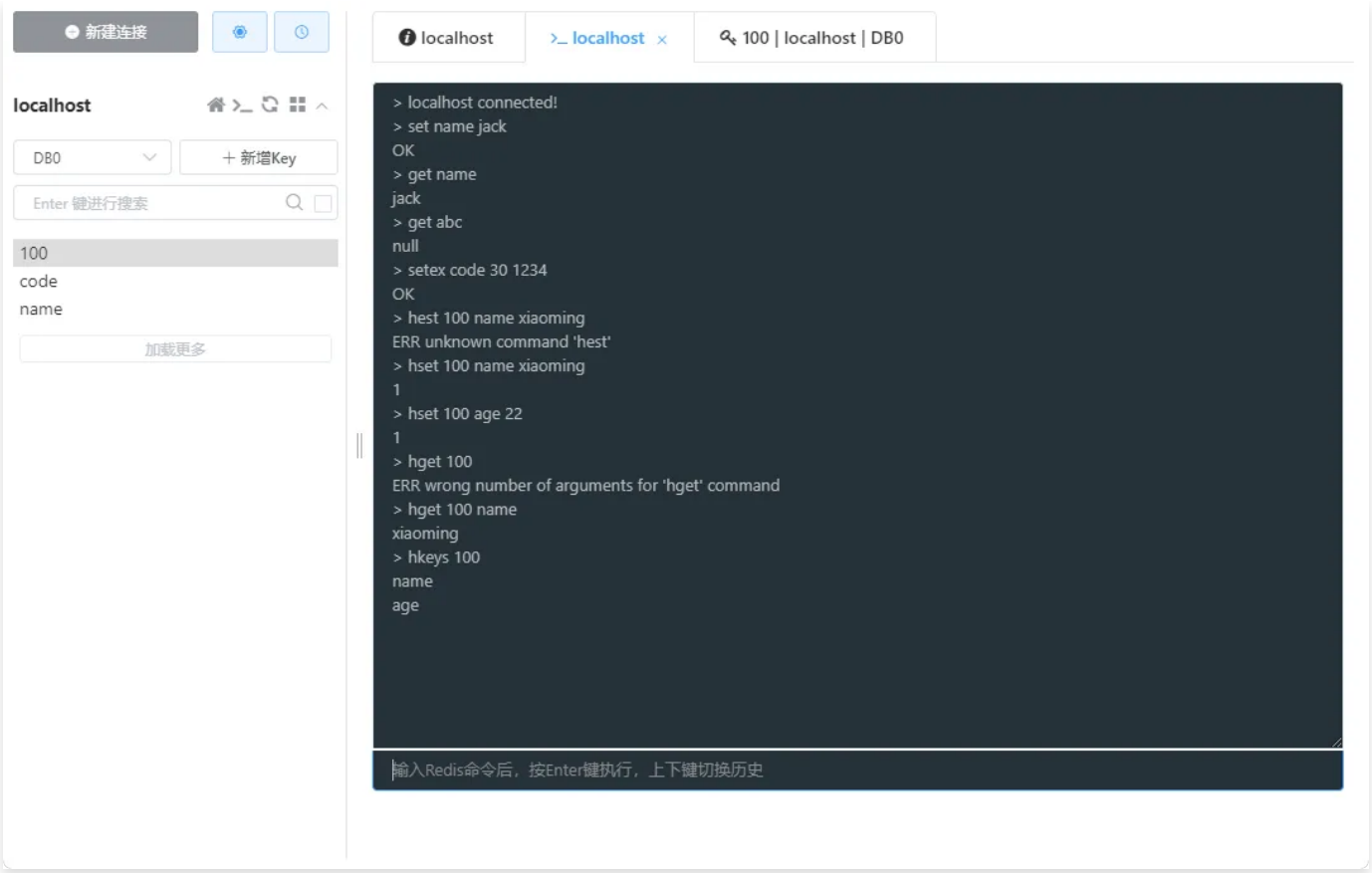
哈希操作命令

Redis hash 是一个string类型的 field 和 value 的映射表，hash特别适合用于存储对象，常用命令：

- HSET key field value　　将哈希表key中的字段field的值设为value
- HGET key field　　获取存储在哈希表中指定字段
- HDEL key field　　删除存储在哈希表中的指定字段
- HKEYS key　　获取哈希表中所有字段
- HVALS key　　获取哈希表中所有值


value
key
field1
value1
field2
value2

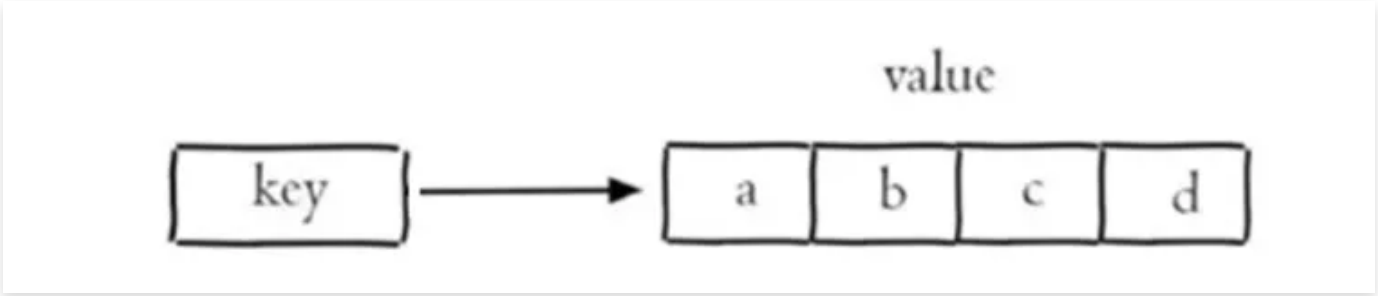

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
name
加载更多
localhost
>_ localhost
100 | localhost | DB0
Hash
100
TTL
-1
添加新行
ID (Total: 2)
Key
Value
输入关键字搜索
1
name
xiaoming
2
age
22
加载更多

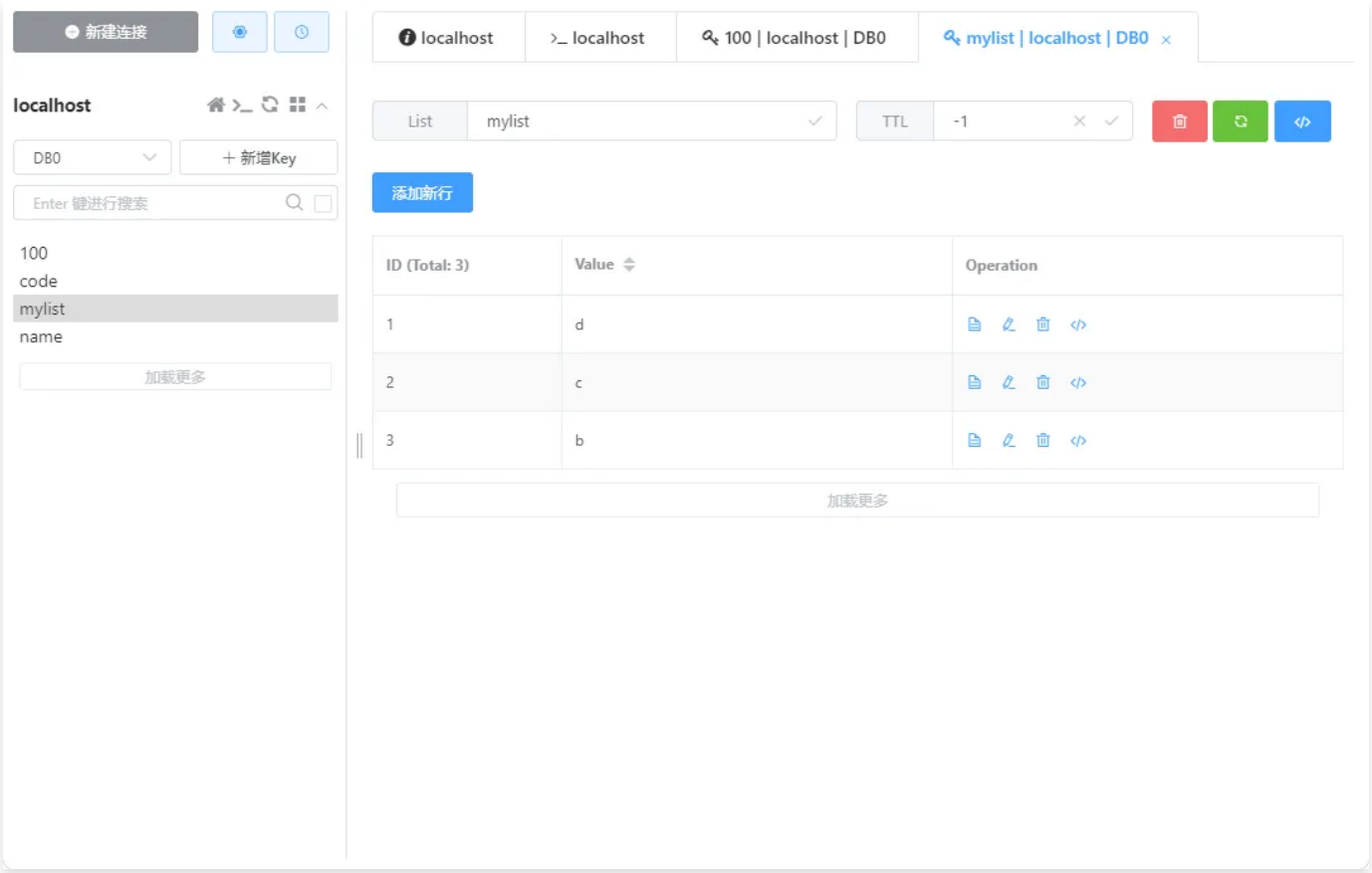


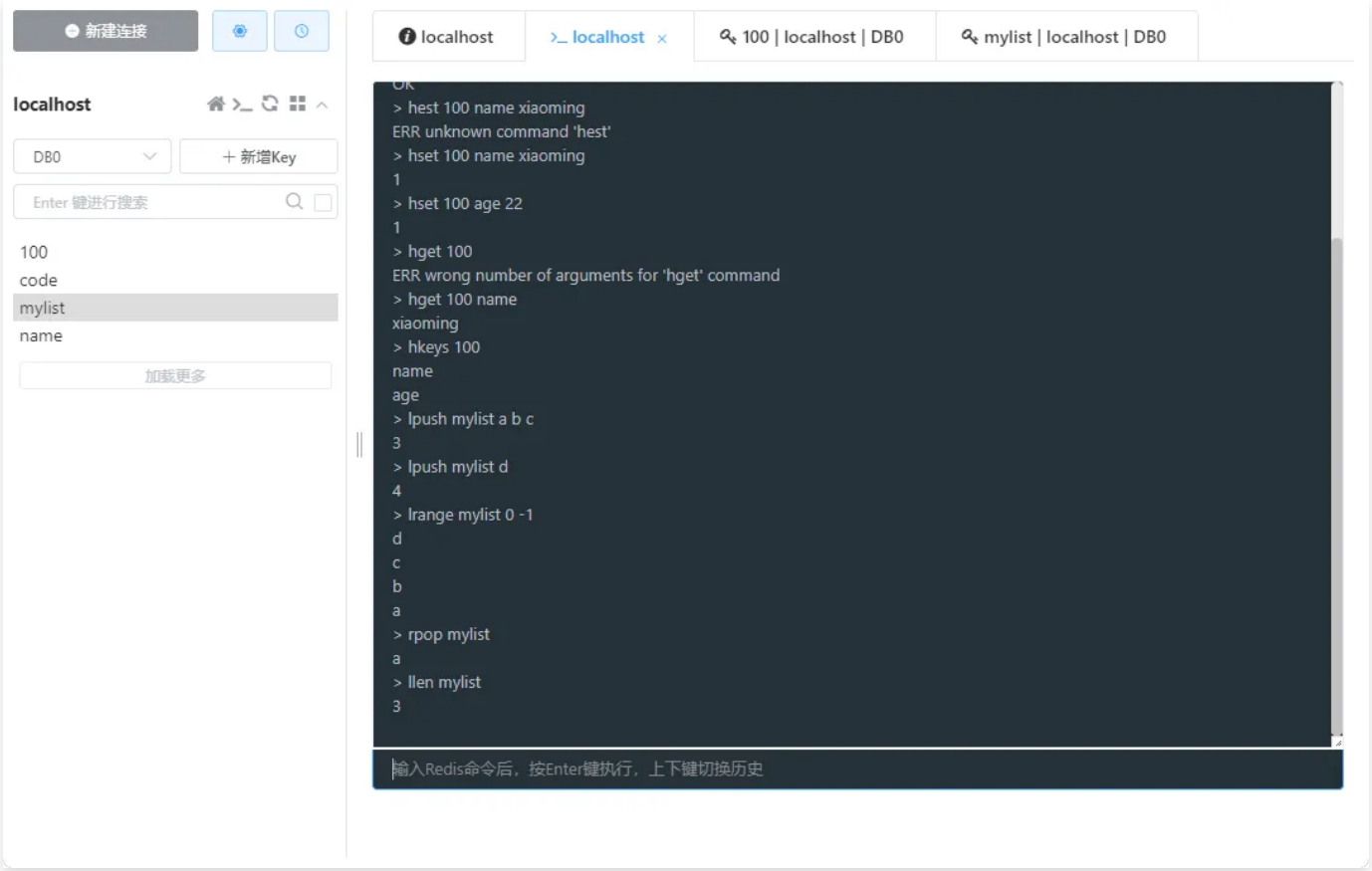
列表操作命令

Redis 列表是简单的字符串列表，按照插入顺序排列，常用命令：

- LPUSH key value1 [value2]　　将一个或多个值插入到列表头部
- LRANGE key start stop　　获取列表指定范围内的元素
- RPOP key　　移除并获取列表最后一个元素
- LLEN key　　获取列表的长度



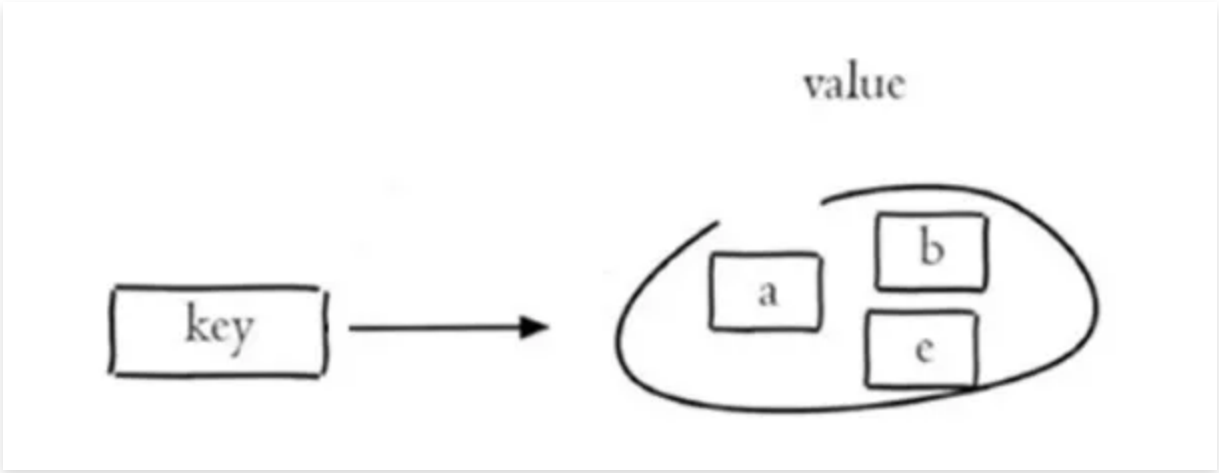

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
mylist
name
加载更多
localhost
localhost
100 | localhost | DB0
mylist | localhost | DB0
List
mylist
TTL
-1
添加新行
ID (Total: 3)
Value
Operation
1
d
2
c
3
b
加载更多

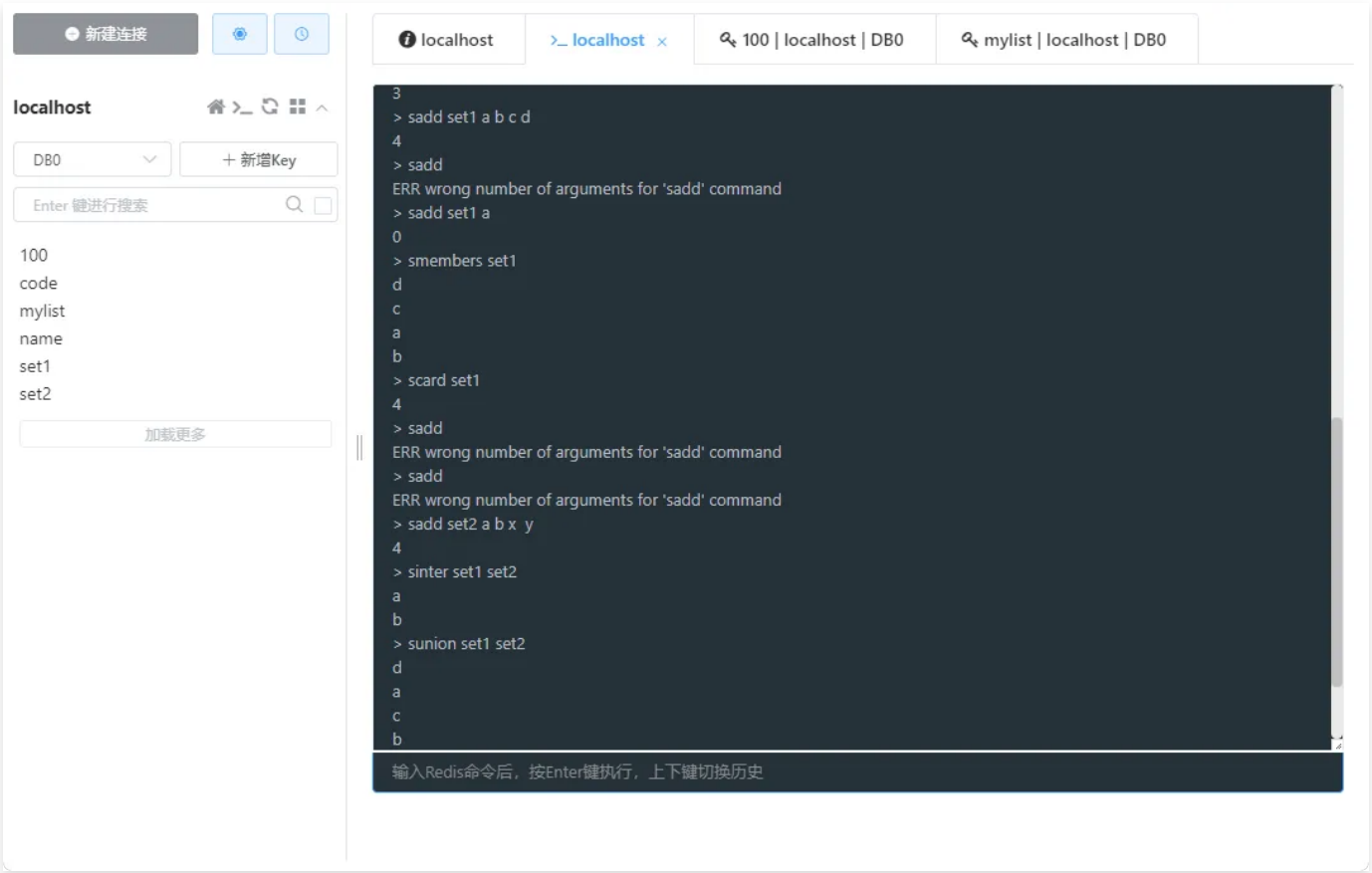

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
mylist
name
加载更多
localhost
localhost
100 | localhost | DB0
mylist | localhost | DB0
OK
> hest 100 name xiaoming
ERR unknown command 'hest'
> hset 100 name xiaoming
1
> hset 100 age 22
1
> hget 100
ERR wrong number of arguments for 'hget' command
> hget 100 name
xiaoming
> hkeys 100
name
age
> lpush mylist a b c
3
> lpush mylist d
4
> lrange mylist 0 -1
d
c
b
a
> rpop mylist
a
> llen mylist
3
输入Redis命令后，按Enter键执行，上下键切换历史

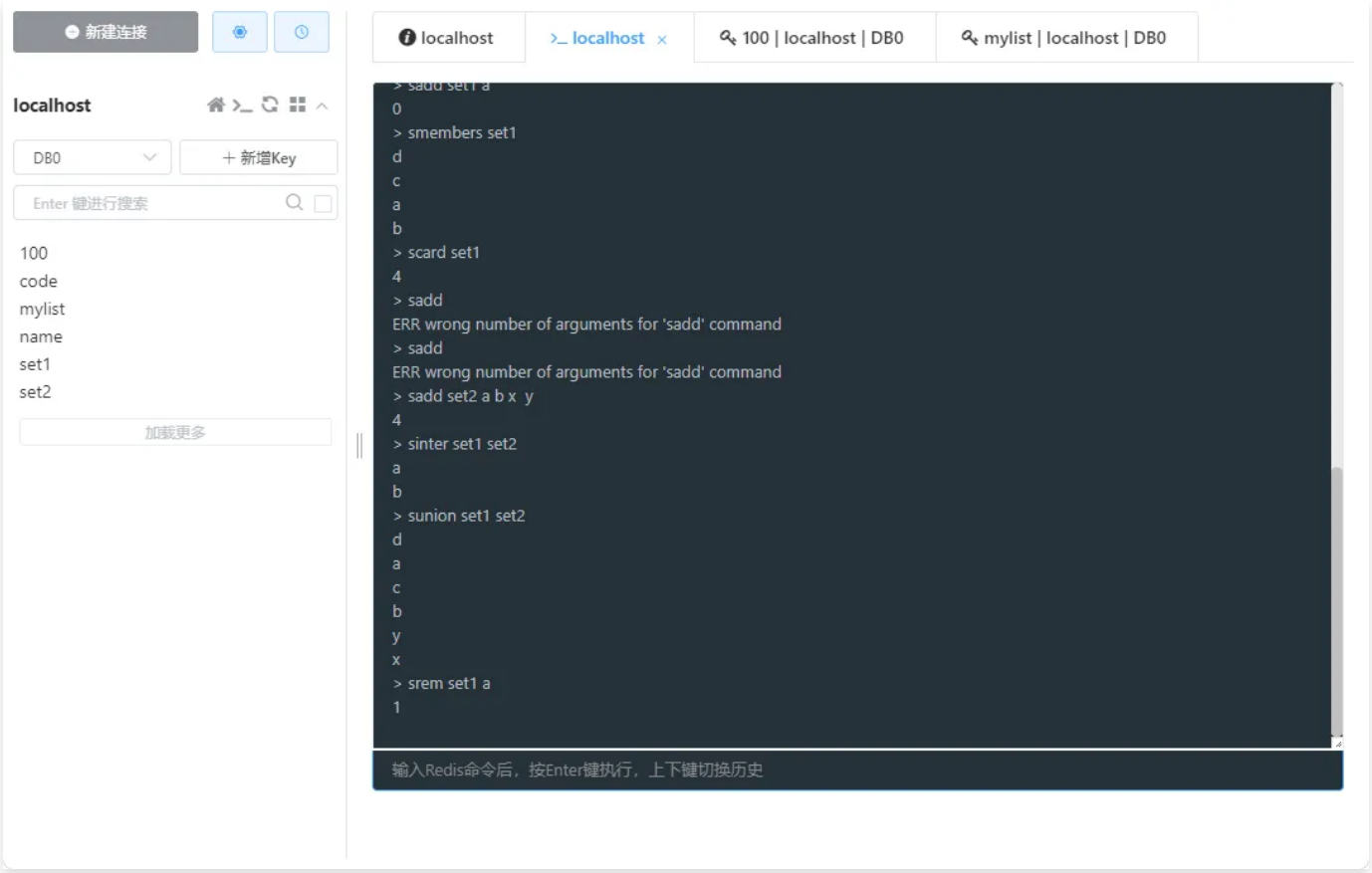
集合操作命令

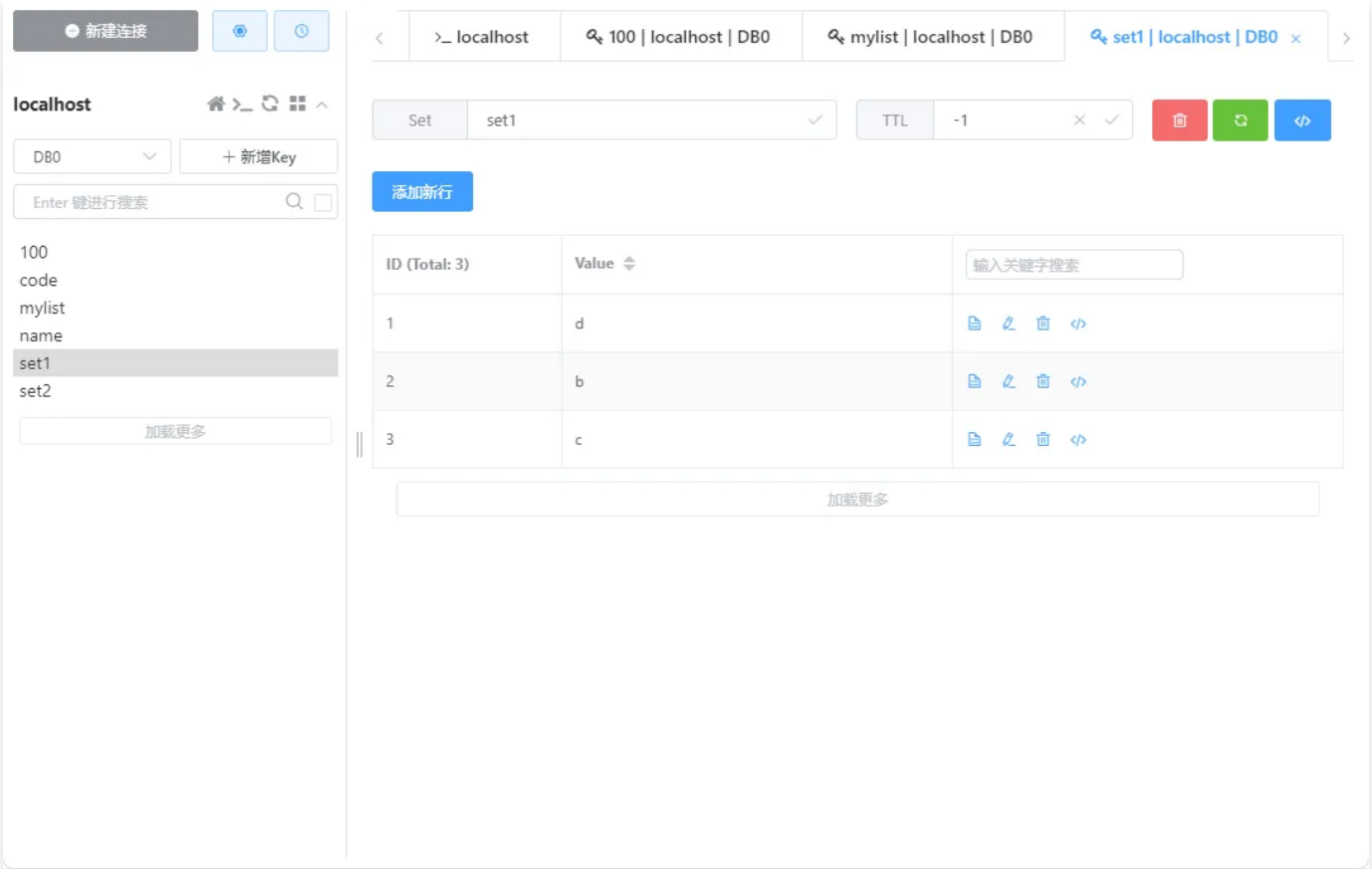
Redis set 是string 类型的无序集合，集合成员是唯一的，集合中不能出现重复的数据，常用的命令：

- SADD key member1 [member2] 向集合添加一个或多个成员
- SMEMBERS key 返回集合中的所有成员
- SCARD key 获取集合的成员数
- SINTER key1 [key2] 返回给定所有集合的交集
- SUNION key1 [key2] 返回所有给定集合的并集
- SREM key member1 [member2] 删除集合中一个或多个成员





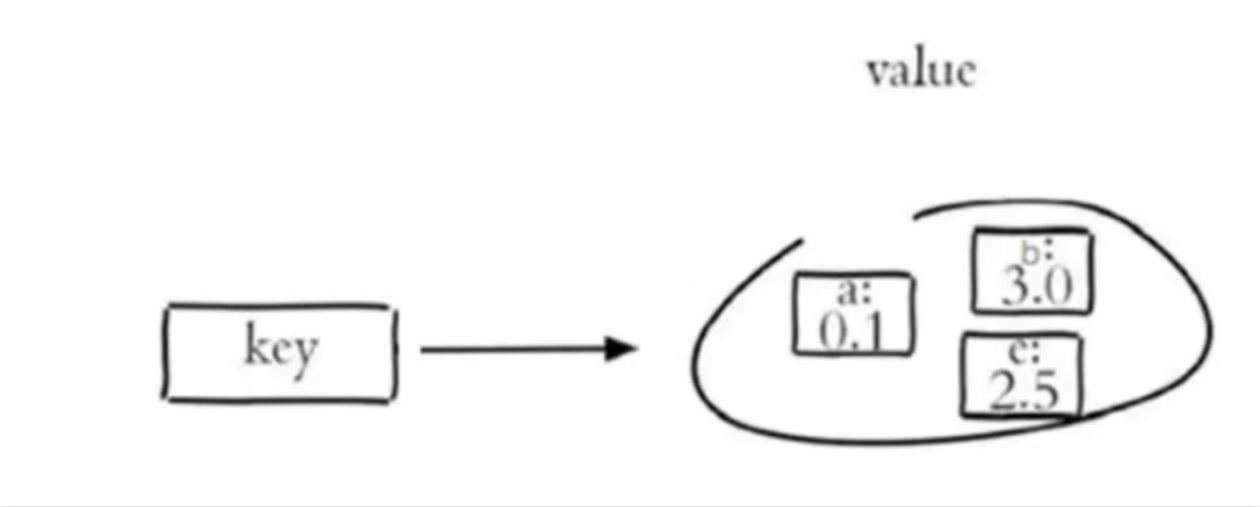

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
mylist
name
set1
set2
加载更多
localhost
localhost
100 | localhost | DB0
mylist | localhost | DB0
0
> smembers set1
d
c
a
b
> scard set1
4
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set2 a b x y
4
> sinter set1 set2
a
b
> sunion set1 set2
d
a
c
b
y
x
> srem set1 a
1
输入Redis命令后，按Enter键执行，上下键切换历史

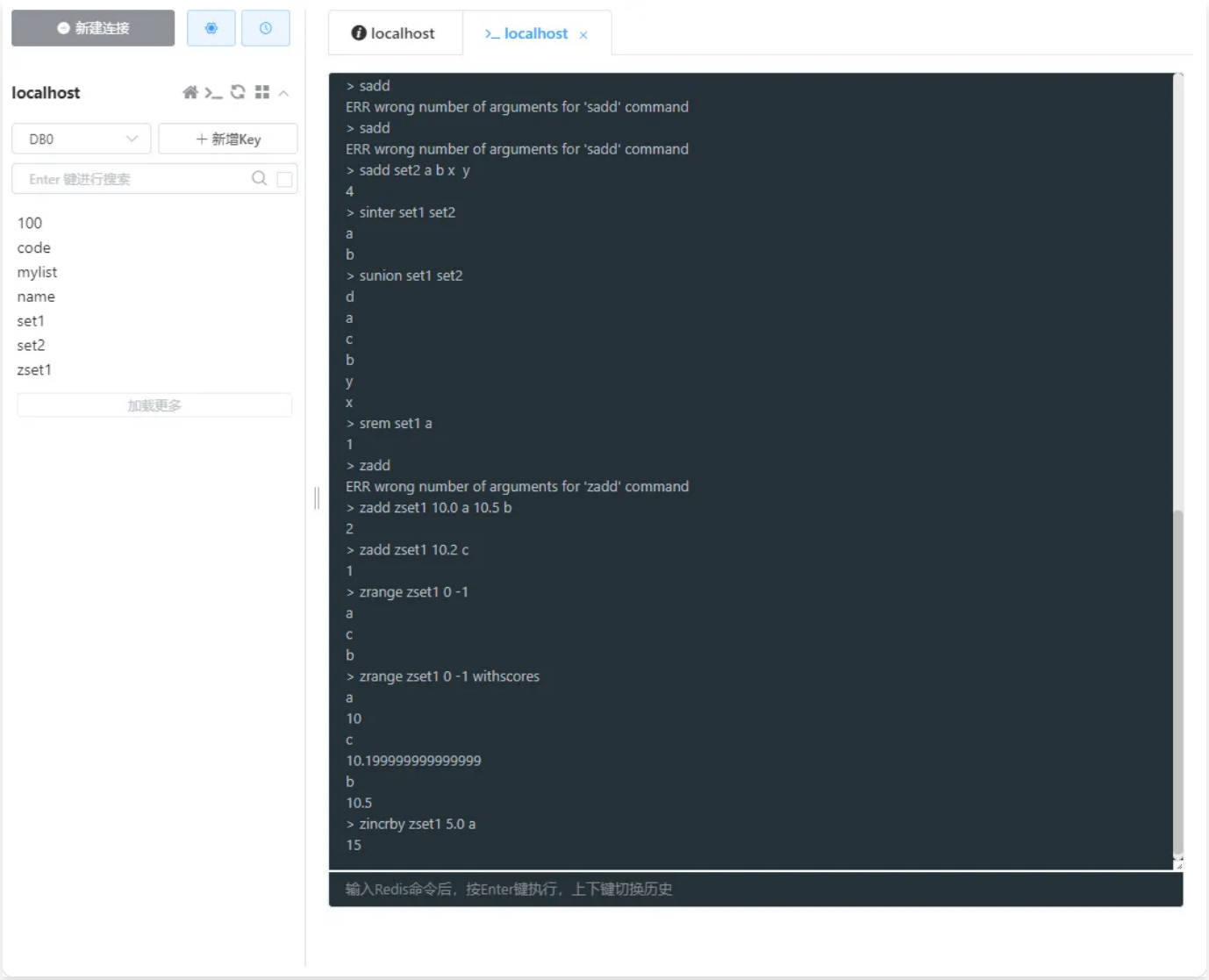

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
mylist
name
set1
set2
加载更多
localhost
100 | localhost | DB0
mylist | localhost | DB0
set1 | localhost | DB0
Set
set1
TTL
-1
添加新行
ID (Total: 3)
Value
输入关键字搜索
1
d
2
b
3
c
加载更多

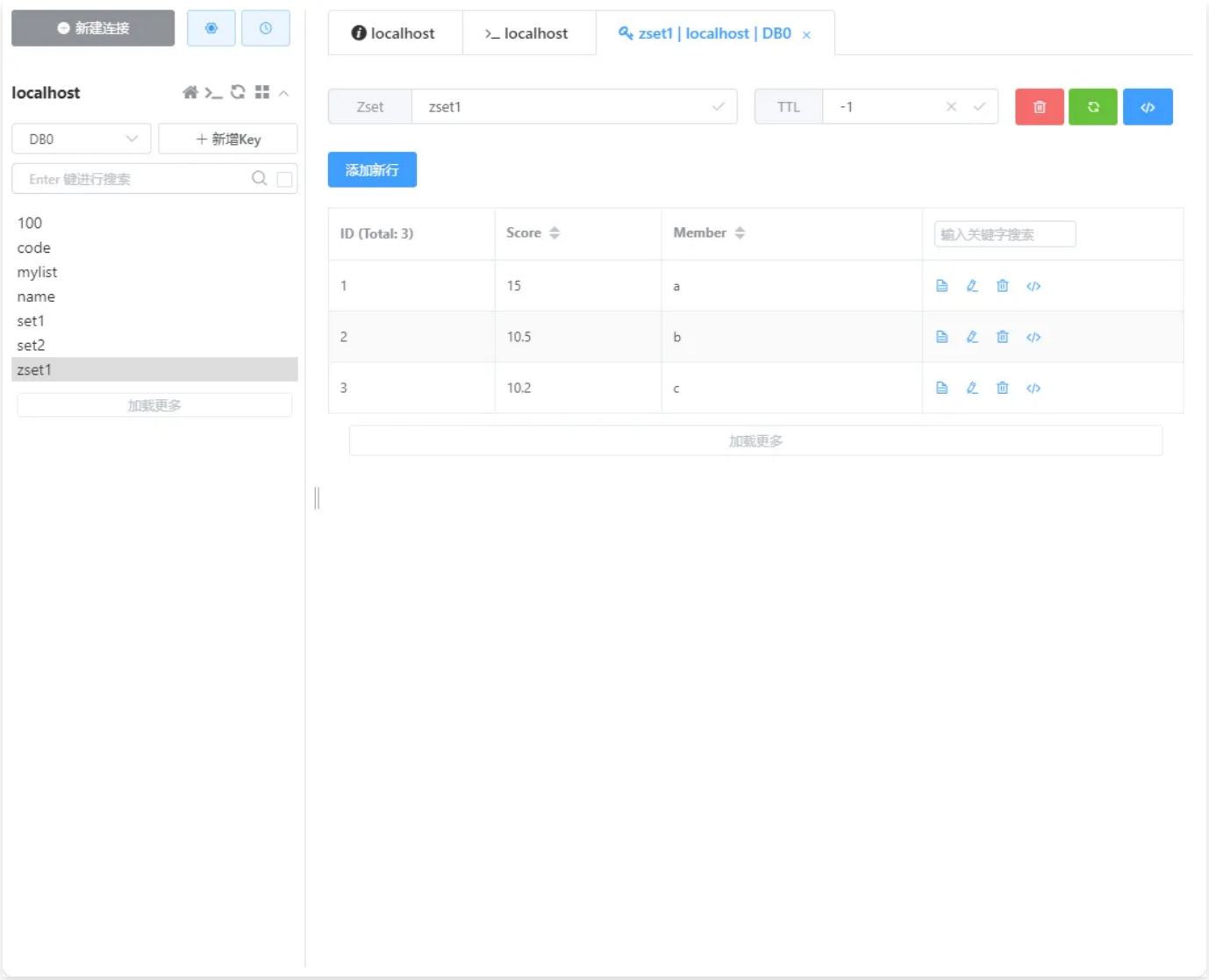
有序集合操作命令

Redis有序集合是string类型元素的集合，且不允许有重复成员，每个元素都会关联一个double类型的分数常用，命令：

- ZADD key score1 member1 [score2 member2]　　向有序集合添加一个或多个成员
- ZRANGE key start stop [WITHSCORES]　　通过索引区间返回有序集合中指定区域内的成员
- ZINCRBY key increment member　　有序集合中对指定成员的分数加上增量increment
- ZREM key member [member ...]　　移除有序集合中的一个或多个成员



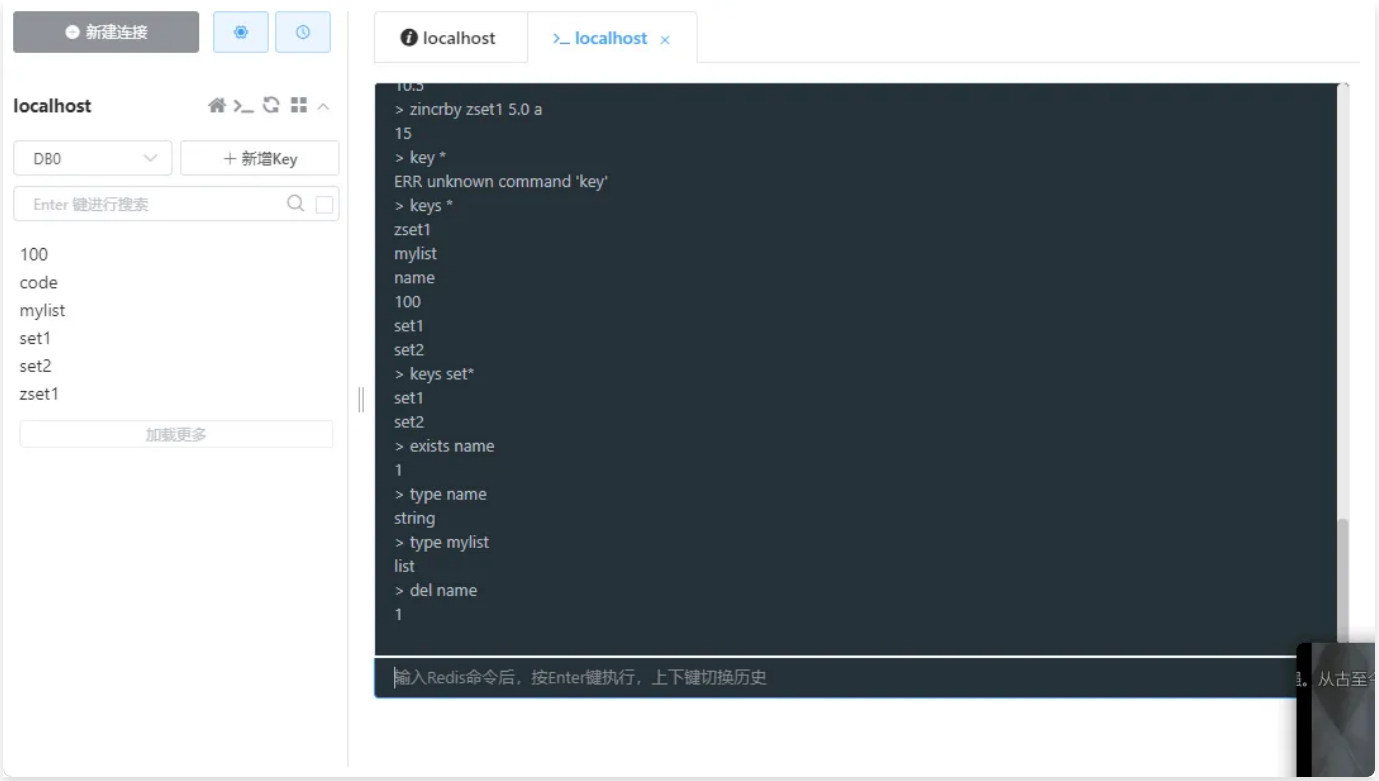

新建连接
localhost
DB0
+ 新增Key
Enter 键进行搜索
100
code
mylist
name
set1
set2
zset1
加载更多
localhost
localhost
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set2 a b x y
4
> sinter set1 set2
a
b
> sunion set1 set2
d
a
c
b
y
x
> srem set1 a
1
> zadd
ERR wrong number of arguments for 'zadd' command
> zadd zset1 10.0 a 10.5 b
2
> zadd zset1 10.2 c
1
> zrange zset1 0 -1
a
c
b
> zrange zset1 0 -1 withscores
a
10
c
10.199999999999999
b
10.5
> zincrby zset1 5.0 a
15
输入Redis命令后，按Enter键执行，上下键切换历史



通用命令

Redis 的通用命令是不分数据类型的，都可以使用的命令：

- KEYS pattern　　查找所有符号给定模式（pattern）的key
- EXISTS key　　检查给定key是否存在
- TYPE key　　返回key所存储的值的类型
- DEL key　　该命令用于在key存在是删除key



在Java中操作Redis

- Redis 的java客户端
- Spring Data Redis使用方式

Redis的java客户端

常见的几种java客户端

- Jedis
- Lettuce
- Spring Data Redis

Sping Data Redis是Spring 的一部分，对Redis底部开发包进行了高度封装。在Spring项目中，可以使用Spring Data Redis来简化操作

Sping Data Redis使用方式

操作步骤：

1. 导入Spring Data Redis 的maven坐标

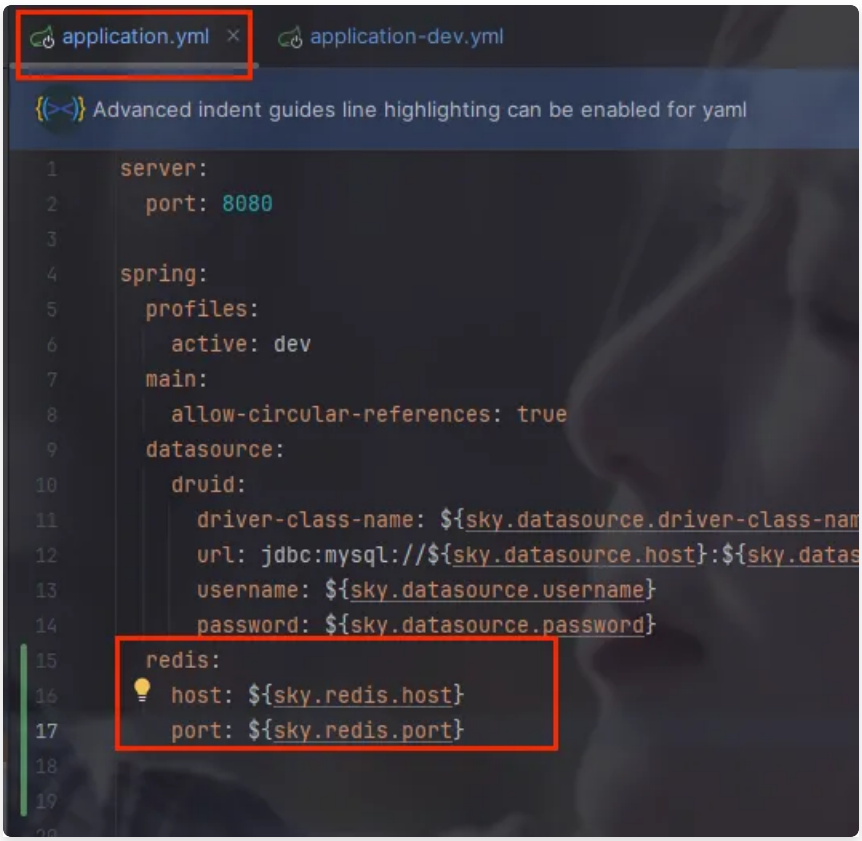
2. 配置Redis数据源
3. 编写配置类，创建RedisTemplate对象
4. 通过RedisTemplate对象操作Redis

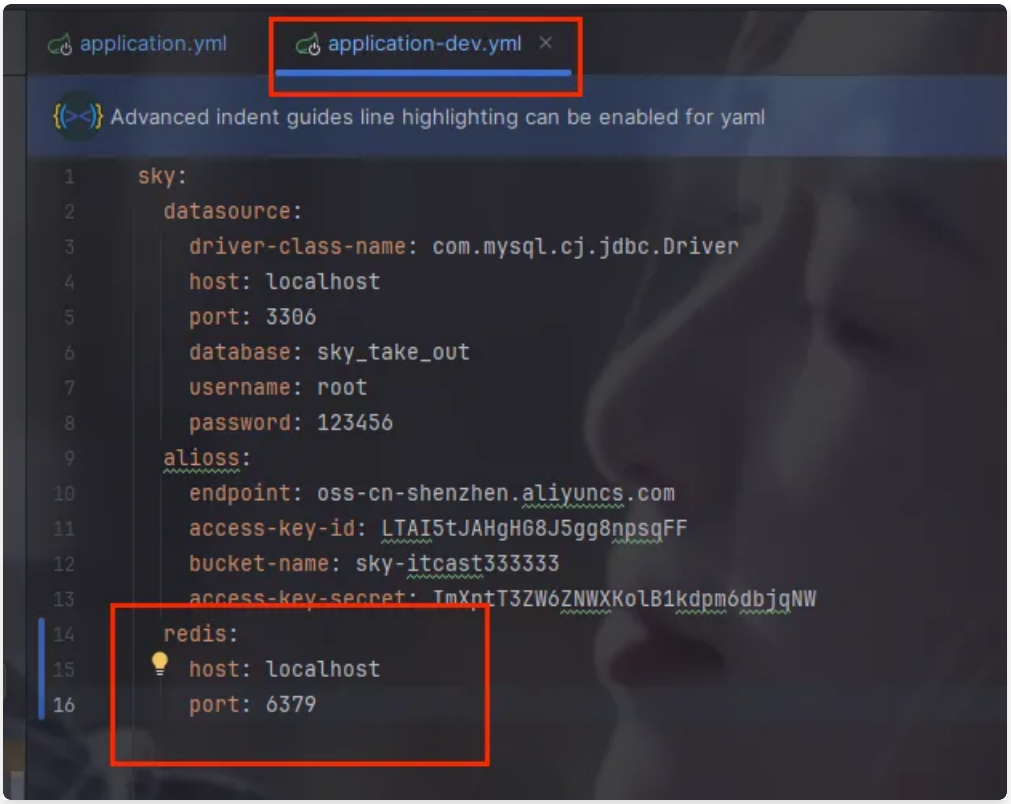
```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

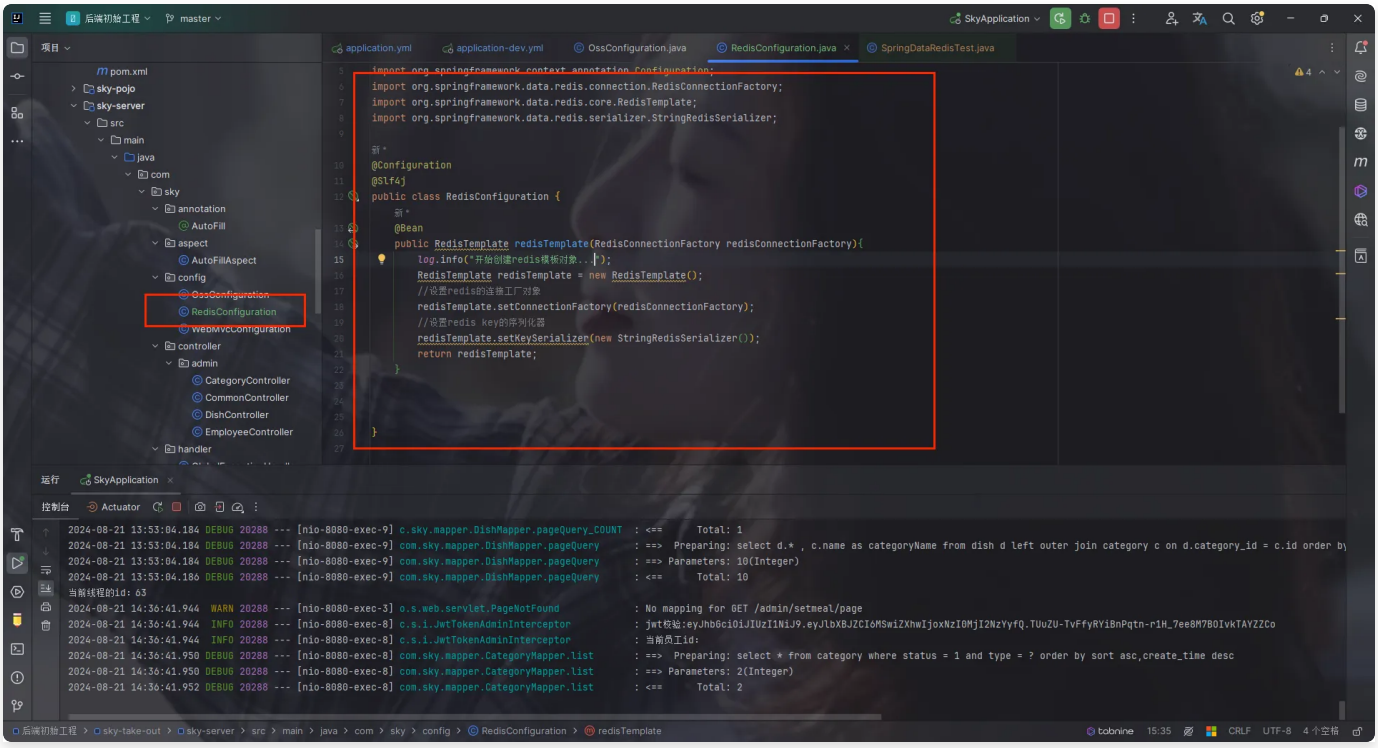
pom.xml

```yaml
spring:
  redis:
    host: localhost
    port: 6379
    password: 123456
```

application.yml




application.yml
application-dev.yml
Advanced indent guides line highlighting can be enabled for yaml
sky:
datasource:
driver-class-name: com.mysql.cj.jdbc.Driver
host: localhost
port: 3306
database: sky_take_out
username: root
password: 123456
alioss:
endpoint: oss-cn-shenzhen.aliyuncs.com
access-key-id: LTAI5tJAHgH68J5gg8npsqFF
bucket-name: sky-itcast333333
access-key-secret: ImXptT3ZW6ZNWXKolB1kdpm6dbjqNW
redis:
host: localhost
port: 6379


application.yml
application-dev.yml
OssConfiguration.java
RedisConfiguration.java
SpringDataRedisTest.java
pom.xml
sky-pojo
sky-server
src
main
java
com
sky
annotation
AutoFill
aspect
AutoFillAspect
config
OssConfiguration
RedisConfiguration
WebMvcConfiguration
controller
admin
CategoryController
CommonController
DishController
EmployeeController
handler
@Configuration
@Slf4j
public class RedisConfiguration {
@Bean
public RedisTemplate redisTemplate(RedisConnectionFactory redisConnectionFactory){
log.info("开始创建redis模板对象...");
RedisTemplate redisTemplate = new RedisTemplate();
//设置redis的连接工厂对象
redisTemplate.setConnectionFactory(redisConnectionFactory);
//设置redis key的序列化器
redisTemplate.setKeySerializer(new StringRedisSerializer());
return redisTemplate;
}
}
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] c.sky.mapper.DishMapper.pageQuery_COUNT : <== Total: 1
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery : ==> Preparing: select d.* , c.name as categoryName from dish d left outer join category c on d.category_id = c.id order by
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery : ==> Parameters: 10(Integer)
2024-08-21 13:53:04.186 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery : <== Total: 10
当前线程的id: 63
2024-08-21 14:36:41.944 WARN 20288 --- [nio-8080-exec-3] o.s.web.servlet.PageNotFound : No mapping for GET /admin/setmeal/page
2024-08-21 14:36:41.944 INFO 20288 --- [nio-8080-exec-8] c.s.i.JwtTokenAdminInterceptor : jwt校验:eyJhbGciOiJIUzI1NiJ9.eyJlbXBJZCI6MSwiZXhwIjoxNzI0MjI2NzYyfQ.TUuZU-TvFfyRYiBnPqtn-r1H_7ee8M7BOIvkTAYZZCo
2024-08-21 14:36:41.944 INFO 20288 --- [nio-8080-exec-8] c.s.i.JwtTokenAdminInterceptor : 当前员工id:
2024-08-21 14:36:41.950 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list : ==> Preparing: select * from category where status = 1 and type = ? order by sort asc,create_time desc
2024-08-21 14:36:41.950 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list : ==> Parameters: 2(Integer)
2024-08-21 14:36:41.952 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list : <== Total: 2

```java
package com.sky.test;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.data.redis.core.*;

import java.util.concurrent.TimeUnit;

@SpringBootTest
public class SpringDataRedisTest {

    @Autowired
    private RedisTemplate redisTemplate;

    @Test
    public void testRedisTemplate() {
        System.out.println(redisTemplate);
        ValueOperations valueOperations = redisTemplate.opsForValue();
        HashOperations hashOperations = redisTemplate.opsForHash();
        ListOperations listOperations = redisTemplate.opsForList();
        SetOperations setOperations = redisTemplate.opsForSet();
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();
    }
    /**
     * 操作字符串类型的数据
     */
    @Test
    public void testString() {
        // set get setex setnx
        redisTemplate.opsForValue().set("city","北京");
        String city = (String) redisTemplate.opsForValue().get("city");
        System.out.println(city);
        redisTemplate.opsForValue().set("code","1234",3, TimeUnit.MINUTES
);
        redisTemplate.opsForValue().setIfAbsent("lock","1");
        redisTemplate.opsForValue().setIfAbsent("lock","2");
    }

        /**
     * 操作哈希类型的数据
     */
    @Test
    public void testHash(){
        //hset hget hdel hkeys hvals
```

```
        HashOperations hashOperations = redisTemplate.opsForHash();

        hashOperations.put("100","name","tom");
        hashOperations.put("100","age","20");

        String name = (String) hashOperations.get("100","name");
        System.out.println(name);

        Set keys = hashOperations.keys("100");
        System.out.println(keys);

        List values = hashOperations.values("100");
        System.out.println(values);

        hashOperations.delete("100","age");
    }
/**
     * 操作集合类型的数据
     */
    @Test
    public void testSet(){
        //sada smembers scard sinter sunion srem
        SetOperations setOperations =  redisTemplate.opsForSet();

        setOperations.add("set1", "a","b","c","d");
        setOperations.add("set2","a","b","x","y");

        Set members = setOperations.members("set1");
        System.out.println(members);

        Long size = setOperations.size("set1");
        System.out.println(size);

        Set intersect = setOperations.intersect("set1","set2");
        System.out.println(intersect);

        Set union =  setOperations.union("set1","set2");
        System.out.println(union);

        setOperations.remove("set1","a","b");
    }

    /**
     *
     */
    @Test
    public void testZSet(){
        //zadd zrange zincrby zrem
```

```
93          ZSetOperations zSetOperations =redisTemplate.opsForZSet();
94
95
96          zSetOperations.add( "zset1","a", 10);
97          zSetOperations.add( "zset1","b",12);
98          zSetOperations.add( "zset1","c",9);
99
100         Set zset1 = zSetOperations.range("zset1", 0,  -1);
101         System.out.println(zset1);
102
103         zSetOperations.incrementScore( "zset1", "c",  10);
104
105         zSetOperations.remove( "zset1","a","b");
106     }
107
108
109     /**
110      * 通用命令操作
111      */
112     @Test
113     public void testCommon(){
114         //keys exists type del
115         Set keys = redisTemplate.keys("*");
116         System.out.println(keys);
117
118         Boolean name = redisTemplate.hasKey("name");
119         Boolean age = redisTemplate.hasKey("set1");
120
121         for(Object key : keys){
122             DataType type = redisTemplate.type(key);
123             System.out.println(type.name());
124         }
125
126         redisTemplate.delete("mylist");
127
128     }
129 }
```